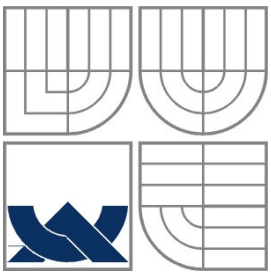
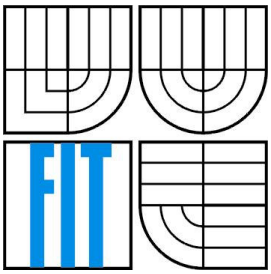


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODUL PRO VÝPOČET MEZD

MODULE FOR SALARY COMPUTATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ POSPÍŠIL

VEDOUCÍ PRÁCE
SUPERVISOR

DOC. ING. JAROSLAV ZENDULKA, CSC.

BRNO 2009

Zadání bakalářské práce/8838/2008/xpospi04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Pospíšil Tomáš**
Obor: Informační technologie
Téma: **Modul pro výpočet mezd**
Kategorie: Databáze

Pokyny:

1. Seznamte se s požadavky na aplikaci mzdové agendy pro malé a střední firmy.
2. Požadavky podrobně analyzujte. Využijte vhodných modelovacích technik.
3. Navrhněte aplikaci pro prostředí PostgreSQL a C#. Rozsah konzultujte s vedoucím bakalářské práce.
4. Navrženou aplikaci realizujte a její funkčnost ověřte na vhodně zvoleném vzorku dat.
5. Zhodnoťte dosažené výsledky

Literatura:

- Marková, H.: Daňové zákony 2008, úplná znění platná k 1.1.2008. 16. vyd. Praha: Grada, 2008. 208s. ISBN: 978-80-247-2385-3.
- Momjian, B.: PostgreSQL : praktický průvodce. 1. vyd. Brno: Computer Press, 2003. 402 s. ISBN: 80-722-6954-2.
- Price, J.: C# : programování databází. Překlad Vilém Vrbický. Praha: Grada, 2005. 623s. ISBN 80-247-0982-1
- Nash, T.: Acceletrated C# 2008. 1st printing. Berkley: Apress, 2007. 510p. ISBN-13: 978-1-59059-873-3.

Při obhajobě semestrální části projektu je požadováno:

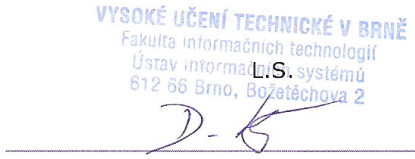
- 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009



L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

V této práci je popsán návrh informačního systému pro výpočet mezd, implementace tohoto modulu a ověření správnosti výsledků. Využívá možností .NET architektury pro informační systém zpracovávající mzdy. Dále je naznačeno propojení SŘBD PostgreSQL s vícevrstvou aplikací pomocí objektově relačního nástroje Entity Framework, který se nově nachází v .NET 3.5 SP1 runtime.

Klíčová slova

.NET Framework 3.5 SP1, C#, Entity framework, PostgreSQL, Vícevrstvá aplikace, ActiveX Data Objects (ADO.NET), Structured Query Language (SQL), Databáze, Objektově relační mapování

Abstract

The thesis deals with design and implementation of an application module for salary computation, its implementation and check validation of results. The program has a form a multi-tier application using PostgreSQL with Entity Framework which is new feature in .NET framework 3.5 SP1.

Keywords

.NET Framework 3.5 SP1, C#, Entity framework, PostgreSQL, Multitier application, ActiveX Data Objects (ADO.NET), Structured Query Language (SQL), Database, Object relation mapping

Citace

Tomáš Pospíšil: Modul pro výpočet mezd, bakalářská práce, Brno, FIT VUT v Brně, 2009

Modul pro výpočet mezd

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. Ing. Jaroslava Zendulky, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Pospíšil
18.5.2009

Poděkování

Rád bych poděkoval doc. Ing. Jaroslavu Zendulkovi, CSc. za věnovaný čas, cenné rady a odborné vedení této práce.

Obsah

1 Úvod

V malých a středně velkých firmách množství zaměstnanců zavdává požadavek na komplexní správu mezd. Používají se k tomu specializované informační systémy, které ve většině případů vzájemně spolupracují a tvoří základ pro ekonomické, správní a personální řízení firmy. V menších firmách se zpravidla nevyplatí celý balík informačních systémů, a to z důvodu velkých finančních nákladů. Častá praxe spočívá v nákupu jednotlivých modulů, kde mzdová část ve většině případech obsahuje základní bloky sestávající z personalistiky, výpočtu mezd podle platné legislativy pro příslušný rok a exporty výstupů do tištěných podkladů pro potřebu evidence. V této práci jsou požadavky na obdobný systém podrobně rozebrány a vhodnými nástroji modelovány. Výstupem je základní blok mzdového modulu pro ověření správnosti postupu.

Motivem této práce byl v době vytváření práce neutěšený stav mzdového systému používaného u podnikatelů v mém okolí. A dále ověření připravenosti prostředí .Net pro vícevrstvou aplikaci spolu s datovým úložištěm PostgreSQL.

Práce je následujícím způsobem strukturována:

- Kapitola 2. pojednává o problematice mezd v českém právním systému platném v roce 2009.
- Kapitola 3. prezentuje .Net architekturu a její využití pro vícevrstvé aplikace v prostředí Windows.
- Kapitola 4. věnuje pozornost RDMS PostgreSQL, použitému jako hlavní datové úložiště informačního systému.
- Kapitola 5. shrnuje požadavky na systém, modelované pomocí případů užití.
- Kapitola 6. obsahuje konceptuální model a na jeho základě navržené schéma databáze.
- Kapitola 7. prezentuje podstatné informace týkající se implementace

Přílohou práce jsou DVD s mzdovým modulem, vývojové diagramy a případy užití.

2 Úvod do problematiky mezd

Právní legislativa České republiky se mění každým rokem několikrát a množství změn nejen v obchodním zákoníku je třeba reflektovat do informačních systémů modelujících dané problémy. Nynější ekonomická situace také zavdává značné množství podnětů na nové zákony, které by snížily daňové zatížení firem i zaměstnanců. Lze tedy očekávat změny, které je nutné v co nejkratším čase implementovat a to z důvodů možných postihů plynoucích z nedodržení zákona. Postup výpočtu mzdy je shodný pro zaměstnavatele jako právnické osoby (dále jen PO) tak i pro zaměstnavatele jako fyzické osoby (dále jen FO).

2.1 Zákoník práce

Zákoník práce (dále jen ZP) [ZAK0] vymezuje pracovně právní vztah zaměstnance vůči zaměstnavateli. V roce 2009 došlo k několika změnám které jsou velmi důležité v návaznosti na výpočet mzdy. Jde především o zrušení daňových pásem a zavedení 15 % rovné daně (obdobně jako v roce 2008), zrušené společné zdanění manželů, zvýšení slev na dani, zavedení stropu pro platbu sociálního a zdravotního pojištění, daňová neuznatelnost pojistného u podnikatelů, zrušení minimální daně, změnu ve zdanění autorských honorářů.

2.1.1 Zaměstnavatel a zaměstnanec

Dle § 7 ZP, se *zaměstnavatelem* rozumí PO nebo FO, která zaměstnává v pracovně právním vztahu fyzickou osobu (zaměstnance). Vystupuje v pracovněprávním vztahu svým jménem a má za tyto vztahy odpovědnost.

2.1.2 Odměna za práci

Odměňování zaměstnanců v pracovním poměru je přímo upraveno v ZP. Dříve se řídilo dvěma samostatnými zákony (zákon o mzdě a zákon o platu).

Podle § 109 ZP, věnujícímu se vymezení pojmů *mzdy* a *platu.* Odměna za práci se dělí do dvou základních kategorií:

- *Mzda* - mzdou jsou odměňování zaměstnanci v nestátních podnicích. Mzdou se rozumí peněžitá plnění, nebo peněžité hodnoty (naturální mzdy) poskytované zaměstnavatelem zaměstnanci za vykonanou práci.
- *Plat* - platem jsou odměňováni zaměstnanci státu
  - územních samosprávních celků (krajů a obcí)

- státních fondů
- příspěvkových organizací (jejichž veškeré náklady jsou hrazeny z rozpočtu zřizovatele)
- školních PO zřízených Ministerstvem školství a tělovýchovy, krajem, nebo obcí

Mzda nebo plat se poskytují dle složitosti, odpovědnosti a náročnosti práce. Dále podle druhu obtížných pracovních podmínek, pracovní výkonosti a odvedených pracovních výsledků. § 141 ZP ustanovuje splatnost odměny ze mzdy nebo platu na datum v následujícím měsíci, který následuje po vzniku práva na mzdu nebo plat.

Dle § 110 ZP je zaměstnavatel povinen za práci stejné hodnoty stanovit zaměstnancům stejnou mzdu či plat. Kde stejnou prací, nebo prací stejné hodnoty se rozumí pracovní výkon obdobné náročnosti ve srovnatelných pracovních podmínkách se srovnatelnou pracovní výkonností a výsledky práce.

Dle § 113 ZP musí být mzda sjednána, stanovena nebo určena před začátkem výkonu práce, za kterou má tato mzda příslušet. Pracovní smlouva, jejímž prostřednictví se zaměstnavatel váže vůči zaměstnanci, musí být písemnou formou. Pokud neobsahuje práva a povinnosti zaměstnance, tak je zaměstnavatel povinen informovat písemnou formou nejdéle do 1 měsíce.

2.2 Pojmy související s výpočtem mzdy

Základní mzda (úkolová nebo časová) – podle počtu vyrobených kusů výrobku, respektive odpracovaných hodin náleží zaměstnanci základní částka, která se dále ponižuje.
Hrubá mzda (dále HM) – tvoří ji základní mzda, mzdové příplatky, náhrada mzdy a ostatní mzdové položky (prémie, odměny stanovené ve mzdových předpisech).
Superhrubá mzda – nově zavedený termín, jedná se součet hrubé mzdy a pojištění placených zaměstnavatelem (sociální a zdravotní). V důsledku se jedná o celkovou částku vynaloženou zaměstnavatelem na mzdu zaměstnanci.
Pojistné na sociální zabezpečení (dále sociální pojištění) – jedná se pojistné na důchodové pojištění, pojistné na nemocenské pojištění a příspěvek na státní politiku zaměstnanosti.
Zdravotní pojištění – druh pojištění, z něhož je hrazena všeobecná zdravotní péče.

2.3 Výpočet mzdy

	Ukázka výpočtu mzdy (pro rok 2009)	
Měsíční hrubá mzda např.: 30 000,00 Kč		Částka
Hrubá mzda (HM)	Součet veškerých položek, které připadají zaměstnanci za práci odvedenou v příslušném kalendářním měsíci. Skládá se ze základní mzdy, příplatků, odměn, náhrad mzdy (dovolená, svátek) a dalších plnění.	30 000,00 Kč
Zdravotní pojištění – zaměstnavatel	Zdravotní pojištění, které za zaměstnance odvádí zaměstnavatel (9 % z HM). Pojistné se zaokrouhlí na celé koruny směrem nahoru.	2 700,00 Kč
Sociální pojištění – zaměstnavatel	Sociální pojištění, které za zaměstnance odvádí zaměstnavatel (26 % z HM v roce 2008, 25 % v roce 2009). Pojistné se zaokrouhlí na celé koruny směrem nahoru	7 500,00 Kč
Superhrubá mzda	Superhrubá mzda, která slouží jako základ pro další výpočty. K hrubé mzdě se připočte zdravotní pojištění zaměstnavatele (9 %) a sociální pojištění zaměstnavatele (25 %). Částka se zaokrouhluje na celé stokoruny nahoru.	40 200,00 Kč
Dílčí základ daně	15 % sazba ze superhrubé mzdy	6 030,00 Kč
Sleva na dani	Uplatnitelnou slevou v roce 2009 je např.: sleva na poplatníka 2 070,00 Kč	2 070,00 Kč
Záloha na daň po slevě	Od dílčího základu daně se odečte sleva (v tomto příkladě je to jen sleva na poplatníka)	3 960,00 Kč
Zdravotní pojištění – zaměstnanec	Zdravotní pojištění, které platí zaměstnanec ze své mzdy (4,5 % z HM). Pojistné se zaokrouhlí na celé koruny směrem nahoru.	1 350,00 Kč
Sociální pojištění – zaměstnanec	Sociální pojištění, které platí zaměstnanec ze své mzdy (8 % z HM v roce 2008, 6,5 % v roce 2009). Pojistné se zaokrouhlí na celé koruny směrem nahoru.	1 950,00 Kč
Čistá mzda	Měsíční hrubá mzda - zdravotní pojištění (zaměstnance) - sociální pojištění (zaměstnance) - daň po slevách = čistá mzda	22 740,00 Kč

3 .NET architektura

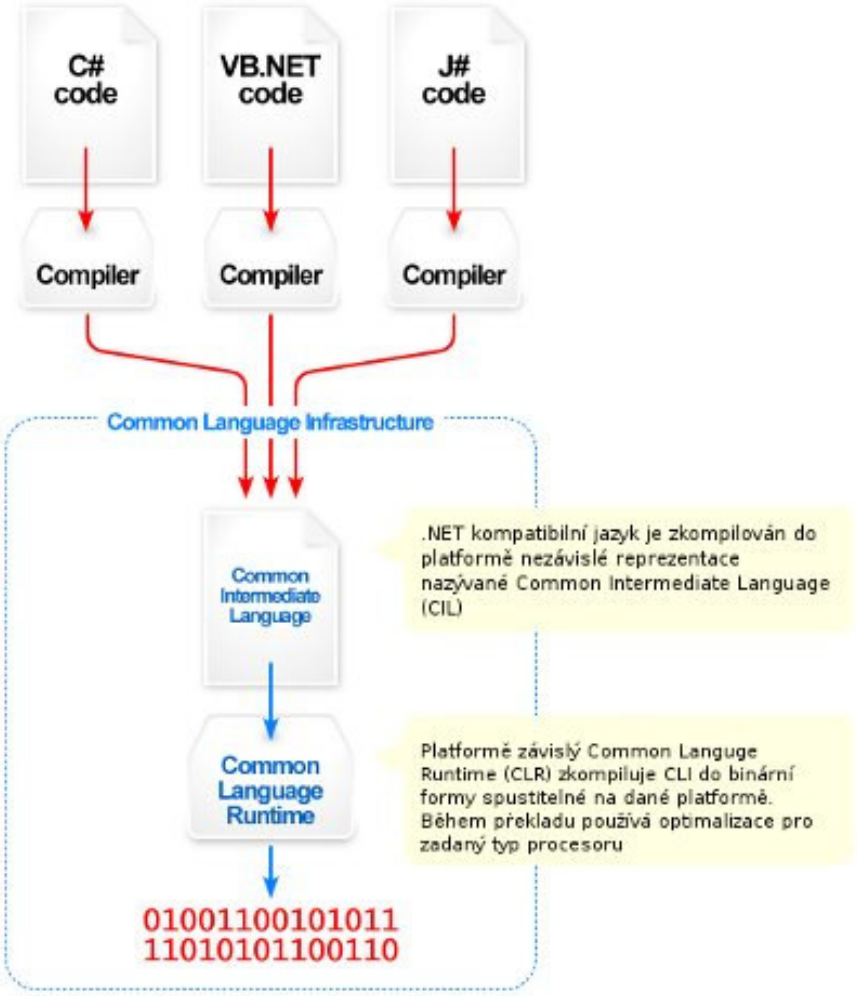
Architektura .Net Framework [NET0] je množina systémových knihoven, které zapouzdřují většinu systémových volání. Obsahuje CLR (Common Language Runtime), která obdobně jako v Javě JIT (Just In Time compiling) překládá aplikaci za běhu z CIL (Common Intermediate Language), který je pak CLR zpracován. Díky tomuto využití virtuálního stroje, lze odstínit různé architektury a provést případné optimalizace až při finálním spuštění na konkrétním procesoru. Framework jako takový je přidáván do Windows od verze Windows Server 2003 a je obsažen nejen operačních systémech pro osobní počítače, ale i v herních konzolách XBOX a mobilních zařízeních s Windows Mobile, kde se využívá .NET Compact Framework.

3.1 Překlad .NET aplikace

Zdrojový kód .NET aplikace může být napsán v různých jazycích, jejich překlad totiž neprobíhá přímo do binární podoby, ale do mezikódu a ten je dále interpretován. Tento mezikód je mezi všemi jazyky shodný a lze tedy vytvářet vzájemně spolupracující knihovny funkcí. Jisté úskalí spočívá různých interpretacích klíčových slov pro datové typy. Například *int* v jazyce Visual Basic, se nemusí schodovat s typem *int* v jazyce C#. Z tohoto důvodu jsou definována klíčová slova int32, popřípadě int64 pro vzájemnou interoperabilitu.

Typický průběh překladu .Net aplikace ilustruje *Obr. 1*. Zdrojový kód je přeložen do Common Intermediate Language. Tento jazyk se podobá vysoko úrovňovému assembleru. CIL kód zabezpečuje zpracování vyjímek a správu paměti (Garbage collector), kód je objektově orientovaný a snadno čitelný viz ukázka Hello World aplikace v CIL [HWL0].

```
.assembly Hello {}
.method public static void Main() cil managed
{
     .entrypoint
     .maxstack 1
     ldstr "Hello, world!"
     call void [mscorlib]System.Console::WriteLine(string)
     ret
}
```



Obr. 1: Průběh překladu zdrojového kódu, převzato z [NET0]

Po přeložení do CIL kódu, probíhá překlad do byte kódu (nazývaného *assembly*), který se za pomocí Common language runtime (CLR) spustí.

3.2 Práce s databázemi v .NET

Operace nad databázemi zajišťuje množina knihoven pod souhrnným názvem ADO .NET [ADO05]. Jedná se o pokračovatele kolekce tříd a rozhraní známého pod jménem ActiveX Data Objects (ADO), které bylo používáno před uvedením .NET Frameworku. Umožňuje běh tzv. odpojených aplikací, které pracují s lokálně uloženými daty z databáze bez aktivního spojení. Architektura je silně podobná javové knihovně JDBC [JDBC] a skládá ze dvou hlavních částí:

- *DataProvider* – třída zajišťující připojení a komunikaci s RDBMS, většina databázových poskytovatelů má svého vlastního providera, který využívá specifické vlastnosti úložiště a umožňuje optimální přístup k němu. Dále existuje obecný poskytovatel, který přes ODBC

přistupuje k databázím. Tento ovšem není optimalizován pro specifické úložiště a neumožňuje využití jeho plného potenciálu.

- *DataSet* – reprezentující kolekci objektů naplněnou daty z databáze, xml souboru, popřípadě jiného strukturovaného úložiště.
    - *DataTable* – reprezentace jednoduché databázové tabulky, obsahuje řádky reprezentované třídou *DataRow* a sloupce *DataColumn*
    - *DataView* – analogicky jako v databázi je zde datový pohled objekt, zajišťující výběr sloupců, filtraci údajů a řazení, pracuje nad *DataTable*.
    - *DataRelation* – modeluje vztahy mezi tabulkami pomocí cizích klíčů
    - *Constraint* – popisuje omezení kladená na vlastnosti tabulky, jako jsou unikátnost, neprázdnost pole. V případě porušení omezení je vyvolána výjimka.

V navržené aplikaci se tento zaběhlý způsob práce s databází na uživatelské úrovni nepoužívá, je plně využito možností Entity Frameworku viz *7 Implementace mzdového modulu*.

3.3 Způsob dotazování LINQ

Pro přístup k datům v databázi lze použít čistě jazyka SQL, který jako jediný poskytuje plnou podporu všech databázově závislých specifik, nebo se dá využít skládání dotazů pomocí Language Integrated Query (LINQ [DAT08]). Výhoda spočívá v jednotném způsobu dotazování jak nad daty v databázi, tak i daty v jiných úložištích (např. XML soubor, kolekce hodnot) a využití integrovaného doplňování IntelliSence ve Visual Studiu [VIS08]. Tento způsob dotazování je velmi často používán v projektu, viz *7 Implementace mzdového modulu*.

3.4 Vlastnosti .NET Frameworku 3.5 SP1

Framework [NET08] byl uvolněn v listopadu roku 2008 a jedná se o jeho nejnovější verzi. Mezi největší přednosti patří ADO.NET Entity Framework, jakožto plnohodnotný objektově relační nástroj spolupracující nejen s Microsoft SQL Server, ale i jinými databázovými poskytovateli. Verze 3.5 SP1 stále vychází z CLR verze 2.0 a je jistou nástavbou, která se jako výchozí instalovaný framework objeví až v nově příchozích Windows Seven. Další vítanou změnou je otevření části zdrojových kódů, které umožňuje snadnější způsob ladění a hledání chyb, ale i nahlédnout, jakým způsobem jsou jednotlivé komponenty implementovány.

3.4.1 Entity Framework

Framework vytváří datovou abstrakci nad daty v databázi a poskytuje konceptuální schéma datového zdroje aplikaci. Opět zde existuje podobnost s obdobným systémem Java Persistence API [JPA06].

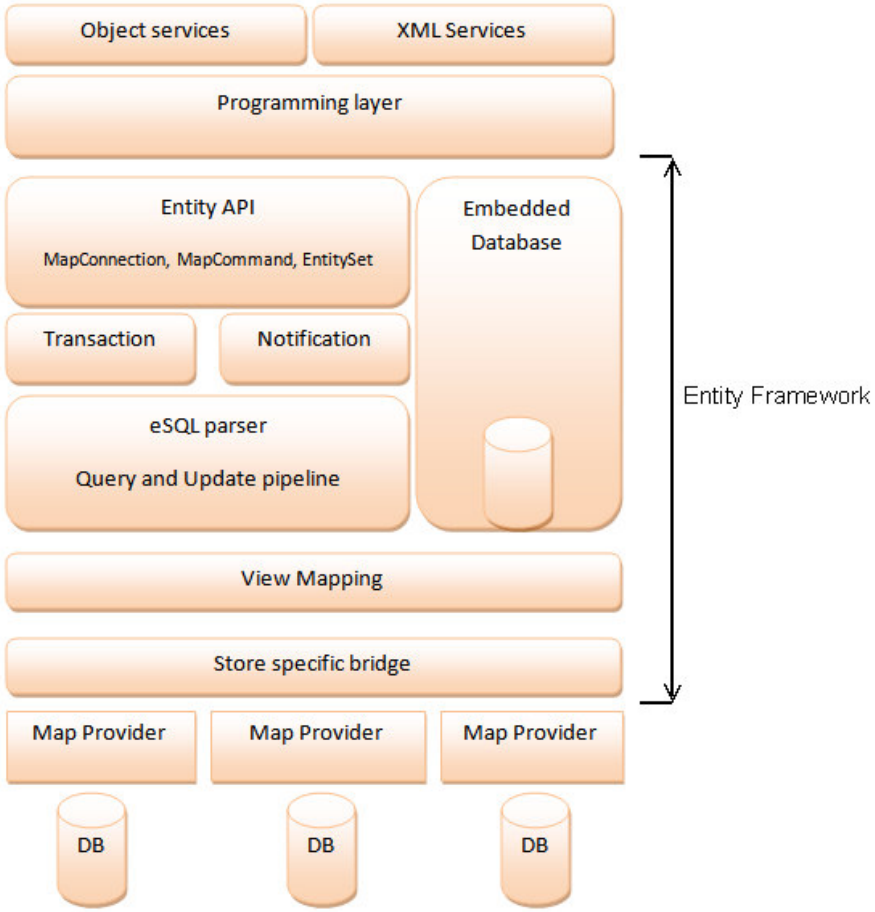
Objektově relační nástroj značně usnadňuje použití datového úložiště hlavně při vytváření dotazů, kdy lze využít znalosti typů jednotlivých položek i znalosti vazeb mezi tabulkami a nechat na frameworku vygenerovat SQL kód pro práci s nimi. Na druhou stranu je nutné zvážit náročnost tohoto řešení, kdy se mezi databázi a aplikaci přidá další vrstva, která musí uchovávat metadata potřebná pro vygenerováním výsledného SQL dotazu. Tato potřebná metadata mohou být problém při velmi rozsáhlé databázi, kdy se prvním voláním dotazu nad databází vyvolá překlad všech závislých vazeb, a tudíž je generována znatelná režie pro celý program.

Podle benchmarku publikovaného v [PEF09] kapitole 16.4 je při prvním volání dotazu pomocí LINQu doba zpracování 10krát delší, oproti původnímu přístupu s ADO .NET. Každý další dotaz je již zpracován stále zhruba 4krát pomaleji. Je to cena za objektový přístup k datům a intuitivnější způsob dotazování. Provádění dotazů lze urychlit jejich překladem při prvním spuštění aplikace, kdy se dotazy přímo přeloží do SQL reprezentace, a důsledným využitím cache. Pro potřeby navrženého programu se tento způsob nevyplatil z důvodu zachování čitelnosti kódu a menšího počtu datových tabulek s vazbami mezi sebou.

Způsob mapování ilustruje *Obr. 2*, kde relační (logické) schéma normalizované databáze je mapováno abstraktní vrstvou na entity, které reprezentují tabulky v databázi. Viz následující ukázka mapování tabulky ekonomických subjektů na typ entity:

```
<EntityType Name="ekonomickesubjekty">
    <Key>
       <PropertyRef Name="ekonomickysubjektid" />
    </Key>
   <Property Name="ekonomickysubjektid" Type="int4" Nullable="false"
StoreGeneratedPattern="Identity" />
    <Property Name="typspolecnostiid" Type="int4" Nullable="false" />
    <Property Name="bankovnispojeniid" Type="int4" />
    <Property Name="statvydejceico" Type="int4" />
    <Property Name="nazev" Type="varchar" Nullable="false"
MaxLength="200"/>
    <Property Name="mesto" Type="varchar" Nullable="false"
MaxLength="150"/>
    <Property Name="okres" Type="varchar" Nullable="false"
MaxLength="250"/>
```

```
    <Property Name="ulice" Type="varchar" MaxLength="150" />
    <Property Name="cislodomu" Type="int4" Nullable="false" />
    <Property Name="psc" Type="varchar" Nullable="false" MaxLength="50" />
    <Property Name="dic" Type="varchar" MaxLength="200" />
    <Property Name="ico" Type="varchar" Nullable="false" MaxLength="50" />
    <Property Name="variabilnisymbolossz" Type="varchar" Nullable="false"
MaxLength="50" />
    <Property Name="vyplatnitermin" Type="int4" />
    <Property Name="datumpridani" Type="timestamp" Nullable="false" />
    <Property Name="sazbazakonnehopojisteni" Type="int4"
Nullable="false" />
    <Property Name="vydelecnacinnostod" Type="date" />
    <Property Name="platnyzaznam" Type="bool" Nullable="false" />
</EntityType>
```



Obr. 2: Abstrakce Entity Frameworku převzato z [EFP08]

Popis základních částí Entity Framework architektury:

- *Map Provider* – který provádí transformaci LINQ a eSQL (způsob objektového dotazování se syntaxí SQL) [SQL09] dotazů do databázově specifických SQL dotazů.
    - *Store specific bridge* – abstrakce pro překlad generického dotazu, na abstraktní databázově specifický dotaz
    - *View Mapping* – mapování databázových pohledů na entity, které jsou pouze pro čtění
- *eSQL parser* – překlad entity SQL dotazů, na abstraktní dotazový strom
- *Entity API* – rozhraní poskytováno pro přístup k vytváření dotazů a nových entit.

4 RDBS PostgreSQL

Systém řízení báze dat PostgreSQL [PGS83] je systém s otevřeným zdrojovým kódem, který je hojně používán v mnoha různých projektech [PGU09]. Z vysokoškolského prostředí se jedná např. o Univerzitu Karlovu či University of California, Berkley. Případně z komerčního světa např. firmy Apple, Sun a jiní. Plně podporuje ACID, transakce, indexaci, obsahuje datové typy z SQL92 i SQL99. Největší předností tohoto systému je jeho otevřenost, lze si nadefinovat vlastní datové typy, pro databázové triggery si můžeme zvolit jazyk PL/pgSQL, který není nepodobný PL/SQL od Oraclu, ale i jiné jako C, Python, Perl. Díky snadné přenositelnosti kódu běží PostgreSQL na Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris) a Windows (32 i 64 bitových) operačních systémech.

Tato otevřenost v licenci ústí v rychlé odezvě na nově vznikající trendy. Tudíž jako jedno z prvních datových úložišť (mimo Microsoft) podporovalo plně Entity Framework, který je využíván v projektu. Dalším důvodem použití je dřívější zkušenost z vývoje webového informačního systému a jeho správy na linuxovém serveru.

5 Požadavky systému mzdové agendy

Požadavky kladené na zpracování mezd jsou v reflexí platné legislativy České Republiky pro rok 2009. Rozsah implementace by byl příliš náročný pro jednotlivce jako bakalářská práce, tudíž jsme po dohodě s vedoucím práce zúžili zadání na implementaci základní části modulu pro výpočet mezd tj. výpočet časové, či úkolové mzdy, výpočet hrubé, superhrubé mzdy, odvodů na sociální a zdravotní

pojištění zaměstnavatele i zaměstnance a celkové částky k výplatě. Neuvažuje se systém nemocenských.

Celkový systém by měl implementovat funkčnost, která se očekává od zpracování mezd v menších a středně velkých firmách v obdobných systémech, které již existují na trhu. Jedná se v nejčastějších případech o MRP Mzdy a personalistika, Pohoda Mzdy a personalistika a jim podobné. Systém musí splňovat požadavky na správu více firem s evidencí příslušných zaměstnanců. Počítá odvody na sociální a zdravotní pojištění, daň ze mzdy a nově i nemocenské dávky zaměstnanci, které se v minulých letech přenechávalo na okresní správě sociálního zabezpečení (dále OSSZ).

Uživatel obsluhující systém očekává obdobné ovládání, na jaké je zvyklý z dosud používaného systému. Je tedy nutné přizpůsobit rozložení prvků na formuláři takovým způsobem, aby uživateli usnadnilo přechod, a eliminovat nedorozumění způsobené jiným významem popisků a akcí přidružených k nim.

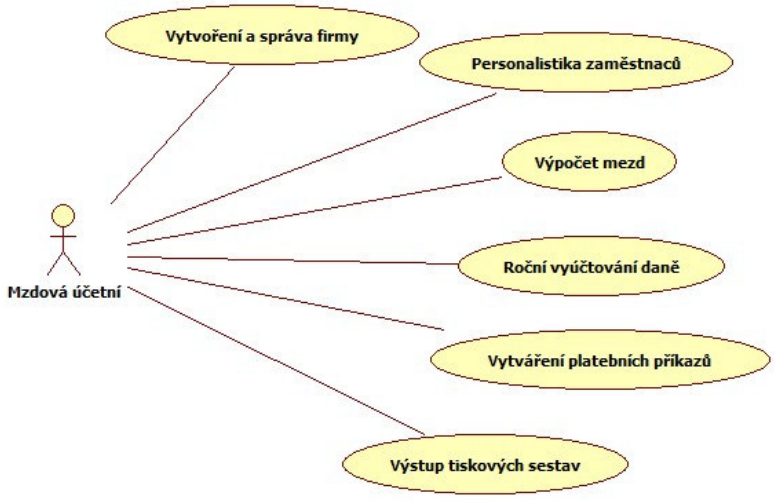
5.1 Případy užití

Jako první fáze před implementací programu bývá často vypracování případů užití viz *Obr. 3*. Jedná se o techniku modelování požadované funkcionality vyvíjeného systému. V knize o modelování případů užití [UC05] jsou v 11. kapitole definovány formáty stylů zápisu, podle níž se držím plně popisného formátu.

Hlavními účastníky, kteří se zapojují do procesu správy mezd jsou: mzdová účetní, zaměstnavatel, zaměstnanec, OSSZ, zdravotní pojišťovna, finanční úřad a pojišťovna zákonné odpovědnosti zaměstnavatele. Každý v určité části zpracování a evidenci mezd hraje svou roli a nelze je nezahrnout do celkového náhledu na systém.

Systém musí implementovat základní funkcionalitu jakou je výpočet mzdové položky pro zaměstnance, kdy si mzdová účetní vyžádá všechny nutné podklady a zanese potřebné údaje. Systém vede personalistiku pro všechny firmy, které se spravují. Mzdová účetní zadává a aktualizuje všechny údaje o firmách i zaměstnancích, také vyplní údaje potřebné k vypočtení mzdy za předchozí měsíc všem zaměstnancům.

Pro přehled uvádím 3 případy užití, zbylé jsou součástí přílohy. Jakožto primárními účastníky procesu výpočtu mzdy byli identifikováni: zaměstnanec, zaměstnavatel a mzdová účetní (obsluhuje informační systém).



Obr. 3: Diagram případu užití

5.1.1 Případ užití: Vytvoření a správa firmy

Firemní údaje je třeba evidovat pro potřeby výstupních sestav. Jedná se především o údaje týkající se evidenčních čísel pro OSSZ, identifikační číslo organizace (dále IČO) a výplatní termín v měsíci.

Tento případ užití je plně implementován v mzdovém modulu.

Hlavní aktéři: Mzdová účetní

Spouštěče: Přijetí nové firmy do správy, změna v údajích spravované firmy

Základní tok událostí:

1. Mzdová účetní zanese do karty firmy potřebné údaje (název, typ společnosti, stát sídla firmy, okres, město, ulici, číslo domu, psč, výplatní termín, IČO, sazbu zákonného pojištění, variabilní symbol OSSZ).

Alternativní toky událostí:

- Mzdová účetní opraví změněný údaj do mzdového modulu.

Vstupní podmínky: Ověřené všechny vstupní údaje týkající se firmy.

Výstupní podmínky: Spravovaná firma je trvale uložena v datovém úložišti.

Body rozšíření:

5.1.2 Případ užití: Personalistika zaměstnanců

Pro potřeb výpočtu je třeba evidovat zaměstnance, jejich osobní údaje jsou relevantní z pohledu výpočtu mzdy a berou se v potaz například při uplatňování slev na daních, přídavků atd.

Tento případ užití je částečně implementován v mzdovém modulu.

Hlavní aktéři: Mzdová účetní

Spouštěče: Přijetí nového zaměstnance, podání výpovědi zaměstnanci, editace osobních údajů zaměstnance

Základní tok událostí:

1. Výběr a otevření firmy.
2. Mzdová účetní založí nový záznam podle informací zjištěných od zaměstnance, nutné údaje jsou: jméno, příjmení, rodné číslo.
3. Systém uloží záznam do trvalého úložiště dat.

Alternativní toky událostí:

- Mzdová účetní při změně personálních údajů, tento stav reflektuje do karty zaměstnance. Systém aktualizuje pouze údaje editovaného zaměstnance, aktualizace se neprojeví na již vypočtených mzdách.
- Mzdová účetní zadá smazání zaměstnance. Systém zaměstnance označí příznakem, ale v systému fyzicky přetrvá.
- Zaměstnavatel propouští zaměstnance. Mzdová účetní v systému vyplní datum ukončení pracovního poměru. Systém vygeneruje zápočtový list.

Vstupní podmínky: Vytvořena alespoň jedna firma.

Výstupní podmínky: Zaměstnanec je korektně uložen v datovém úložišti se všemi vyžadovanými položkami.

Body rozšíření:

5.1.3 Případ užití: Výpočet mezd

Zaměstnanci po odvedené stanovené práci náleží mzda a ten ji ve výplatní den požaduje. Mzdový modul ze zadaných vstupních údajů vypočítá náležející mzdu a připraví tiskopis (výplatní lístek).

Tento případ užití je částečně implementován v mzdovém modulu.

Hlavní aktéři: Mzdová účetní

Spouštěče: Výplatní den

Základní tok událostí:

1. Mzdová účetní doplní docházku a splněné úkoly.
2. Systém vypočte mzdy všech zaměstnanců se zadanými údaji.
3. Systém spočítá odvody na zdravotní pojišťovnu a připraví daný tiskopis.
4. Systém spočítá odvody na sociální pojištění a připraví daný tiskopis.

5. Systém spočítá daň ze mzdy.

Alternativní toky událostí:

- Systém při výpočtu mezd zjistí chybějící, popřípadě nesprávně vyplněné údaje. Systém zobrazí hlášení s chybou a mzdová účetní chybné údaje opraví. Systém se znovu pokusí mzdu vypočíst.

Vstupní podmínky: Definován výplatní den u vybrané firmy.

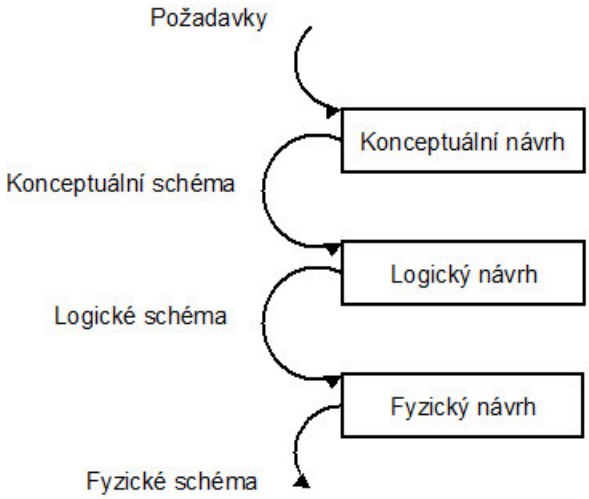
Výstupní podmínky: Vypočtené mzdy spolu s informacemi relevantními k výpočtu trvale uložené v datovém úložišti.

Body rozšíření:

6 Návrh relační databáze

V průběhu návrhu databáze se prochází několika fázemi analýzy viz *Obr. 4.* V průběhu tvorby se konkretizují požadavky do schématu úložiště.

V první fázi se provádí prostřednictvím konzultací s uživateli systému sběr neformálních požadavků, které systém musí splňovat. Takto získaná data jsou základní doménou dat, která je třeba trvale uchovávat, ale která nenesou informaci o datových typech a nejsou normalizována. Vazby mezi nimi jsou popsány z uživatelova pohledu, nikoliv z pohledu vazby mezi tabulkami (jak budou ve finální databázi realizována).



Obr. 4: Průběh návrhu databáze

Konceptuální schéma – je reprezentováno ER diagramem, který definuje statické datové členy a vazby mezi nimi, dále jsou určeny kardinality těchto vztahů. Diagram neobsahuje definice primárních a cizích klíčů, dále nemusí obsahovat atomické atributy. Nutné je identifikovat nezávislé entity, vztahy mezi nimi a závislé entity.

Logické schéma – vytváří relační model z ER diagramu. Relační schéma, obsahuje již definice primárních klíčů, cizích klíčů, datových typů a je vhodné, aby bylo normalizované.

Fyzický návrh – transformuje relační model na přesnou definici domén jednotlivých atributů, který je specifický pro daný RDBMS. Výstupem bývá nejčastěji SQL skript s definicemi integritních omezení, indexů, přístupových metod atd.

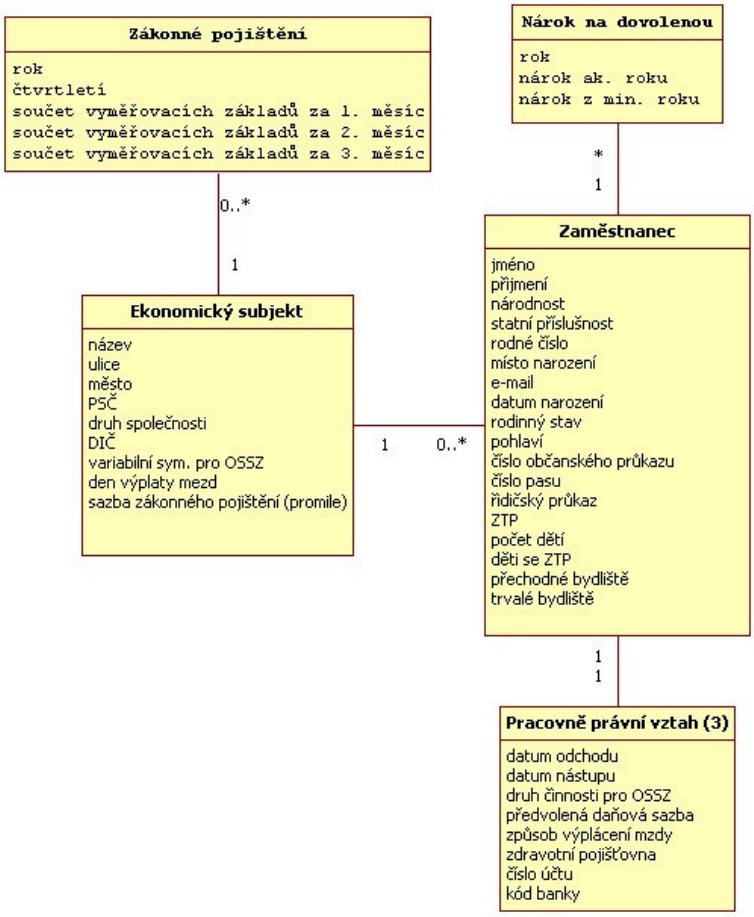
6.1 ER diagram

Entity relationship diagram nemá ustálenou grafickou reprezentaci. Používají se různé způsoby, které vychází z UML. V další části práce je použita notace v takové formě, jaká je přednášena v předmětu IDS.

Celý vypracovaný ER diagram je součástí přílohy C, dále jsou popsány základní části diagramu.

6.1.1 Entity ekonomického subjektu a zaměstnance

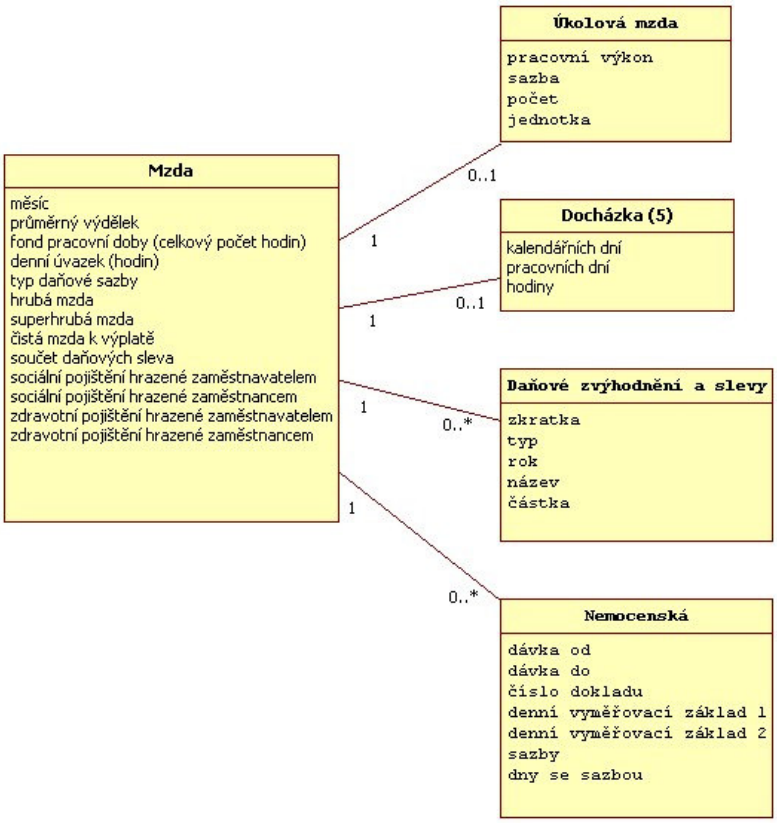
Na diagramu *Obr. 5* je znázorněna vazba *ekonomický subjekt – zaměstnanec*. Kde ekonomický subjekt zaměstnává 0 .. N zaměstnanců. U ekonomického subjektu existuje vazba na entitu *zákonné pojištění*, která vzniká u každého zaměstnavatele v první den pracovně právního vztahu. Entita *Platební příkaz* reprezentuje informaci o platbách provedených ekonomickým subjektem ve prospěch zaměstnanců a státních zařízení vystupujících v pracovně právním vztahu. U zaměstnance je vazba s kardinalitou 1 .. 1 na entitu reprezentující *pracovně právní vztah*, který definuje základní údaje z pracovní smlouvy. Dále u zaměstnance figuruje vazba s kardinalitou 1 .. N vůči entitě *nárok na dovolenou* zaznamenávající průběh čerpání dovolené v průběhu roku a nárok na čerpání dovolené z minulého roku.



Obr. 5: Část ER diagramu zobrazující entity ekonomického subjektu a zaměstnance

6.1.2 Entita mzda a její náležitosti

Na diagramu *Obr. 6* je znázorněna vazba entity *mzda* spolu s jejími náležitostmi důležitými pro výpočet. Mezi entitou *mzda* a *zaměstnanec* existuje také vazba, viz *příloha C* s celkovým ER diagramem. Každá mzda musí uchovávat atributy s vypočtenými hodnotami (hrubá mzda, superhrubá mzda, čistá mzda k výplatě, ...). Na první pohled by se mohlo zdát nedodržení správného stylu návrhu, ale tyto atributy mají své opodstatnění. V první řadě je třeba uchovávat vstupní hodnoty v průběhu výpočtu, pro potřeby právní povinnosti a prokazatelnosti správnosti výpočtu. Dále jsou nutné v případě změny legislativy, a tudíž i změny způsobu určení mzdy při zachování průběhu toku výpočtu.



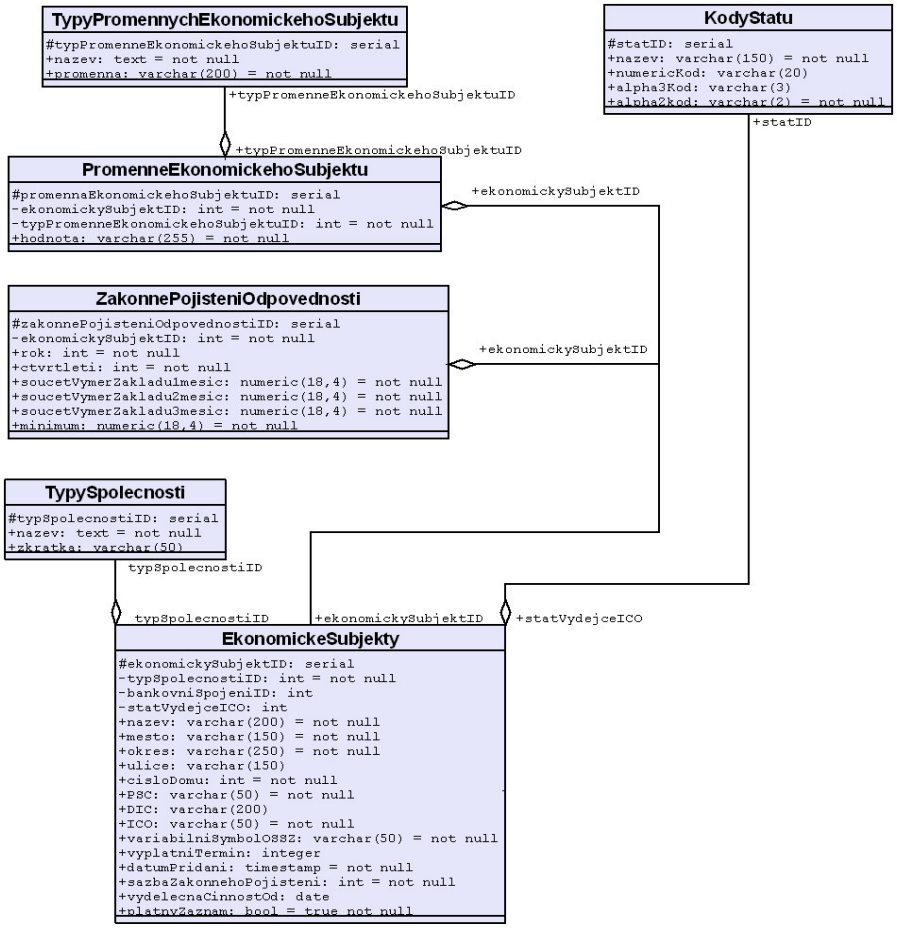
Obr. 6: Část ER diagramu zobrazující entitu mzda a její náležitosti

Entita *mzda* má vazbu na entitu *úkolová mzda* i *docházka* s kardinalitou 0 .. 1, která značí možnost zadání jak časové, tak i úkolové mzdy současně i výlučně. Dalšími navázanými entitami jsou *daňové zvýhodnění a slevy*, *nemocenská* které svou kardinalitou implikují nevynucenou přítomnost.

6.2 Relační schéma

Pro potřeby prezentace bylo z ER diagramu vytvořeno relační schéma, reprezentující strukturu tabulek, jejich vazeb (pomocí cizích klíčů) a datových typů. Pro názornost je entita ekonomického subjektu (popsaná v předchozí kapitole) prezentována po převodu *Obr. 7.* Celkové relační schéma je součástí *přílohy D*.

Převod byl prováděn pomocí programu DIA, který není primárně k těmto operacím určen, a také díky absenci ustáleného značení bylo použito existující vycházející z UML notace. Značka '#' u atributu implikuje primární klíč, značka '-' cizí klíč. Dále je použito pojmenovaných vazeb agregace pro značení spojení tabulek, kde popisek u začátku a konce vazby znamená cizí, respektive primární (popřípadě jiný unikátní klíč).



Obr. 7: Část relačního schématu zachycující tabulky s přímou vazbou na ekonomické subjekty

Neatomické položky jako typ společnosti a stát výdejce IČO byly převedeny na samostatné tabulky. Další návazné tabulky jako číselník typů proměnných ekonomického subjektu a proměnné ekonomického subjektu slouží jako pomocné pro potřeby výpočtu mzdy, popřípadě výstup tiskových sestav. Nejsou nezbytné, proto se nenachází v ER diagramu.

7 Implementace mzdového modulu

Primární platformou pro vývoj se staly Windows, protože ve firemním prostředí nelze očekávat vysokou penetraci Unix-like systémů a u obsluhujících uživatelů je pravděpodobná dřívější zkušenost s programy tohoto druhu. Program se tedy snaží držet typické struktury ovládání s drobným uživatelským vylepšením, které by mělo přispět k zpřehlednění a intuitivnosti. Rozsah pokrývá případy užití popsané v *5.1 Případy užití.* Jmenovitě je implementována funkčnost správy firem

a zaměstnanců, výpočet časové a úkolové mzdy spolu s přesčasy a dovolenými. Výpočet sociálního pojištění placeného zaměstnavatelem i zaměstnancem, výpočet zdravotního pojištění zaměstnavatele i zaměstnance, zdanění mzdy a zákonné pojištění odpovědnosti zaměstnavatele.

7.1 Architektura programu

V programovém modulu (viz *příloha A*) byla důsledně implementována architektura vícevrstvých aplikací [NTIER0]. Vícevrstvá architektura odděluje jednotlivé logické bloky do samostatné vrstvy viz *Obr. 8*. Výhoda takovéhoto řešení spočívá ve velké míře škálovatelnosti a znovupoužitelnosti. Při kolektivní práci umožňuje programátorovi nezávislý vývoj na ostatních částech projektu. Pro tyto výhody byl použit i při implementaci mzdového modulu.

Obr. 8: Schéma vícevrstvé aplikace

Jednotlivé vrstvy jsou v projektu realizovány jako samostatné projekty, kdy jejich výstupem je dynamická knihovna použitá v nadřazené vrstvě. Přístup k databázi je zajištěn pomocí datového poskytovatele Npgsql [NPG09], datová abstraktní vrstva pomocí Entity Framework [EFP08].

7.2 Abstraktní datová vrstva

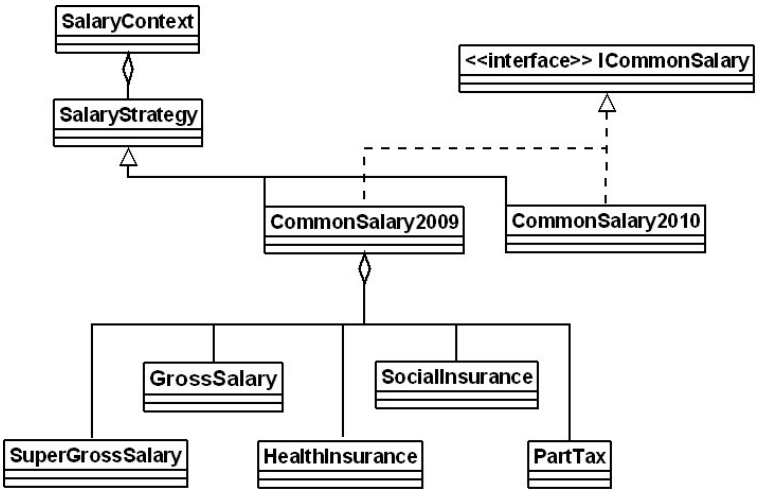
Ve jmenném prostoru DataAbstractionLayer.SalaryDAL se nachází třídy pro práci s daty v objektově ralačním podání. Nástrojem Edmgen2 [EDM02] byly ze schématu databáze reprezentované relačním schématem uvedeným v *příloze D* vygenerovány následující soubory (které v kontextu *Obr. 2* definují *Storage specific bridge*):

- *SalaryDAL.ssdl (storage schema)* – identifikuje tabulky, vazby a omezení v databázi

- *SalaryDAL.msl (map schema)* – provádí mapování mezi entitami a tabulkami, kde lze využít typických rysů OOP jako je dědičnost, zapouzdření. Přiřazuje atributům entit ekvivalenty typů, jež jsou definovány v databázi (např. varchar(200) => String)
- *SalaryDAL.csdl (conceptual schema)* – definuje rozhraní pro práci s entitami, drží kontext objektu při jeho editaci, mazání a vytváření.

7.3 Business logika

Z důvodu budoucího vývoje a snadnější reakce na změny ve výpočtu mezd, byl využit návrhový vzor strategie tak, jak je prezentován v [CDP08]. Při důsledném programování vůči rozhraní nikoliv vůči třídě, bylo docíleno znovupoužitelnosti částí kódu, které se změna legislativy nebude týkat. Viz *Obr. 9*, který ve zjednodušené formě ilustruje implementovaný výpočet. Pro přehlednost byly vynechány rozhraní všech tříd agregovaných do *CommonSalary2009*, tak i obdobná agregace u *CommonSalary2010*, jež reprezentují třídy realizující výpočet mzdy v zadaných obdobích. V práci je implementována třída *CommonSalary2009*.



Obr. 9: Diagram tříd při implementaci výpočtu mzdy

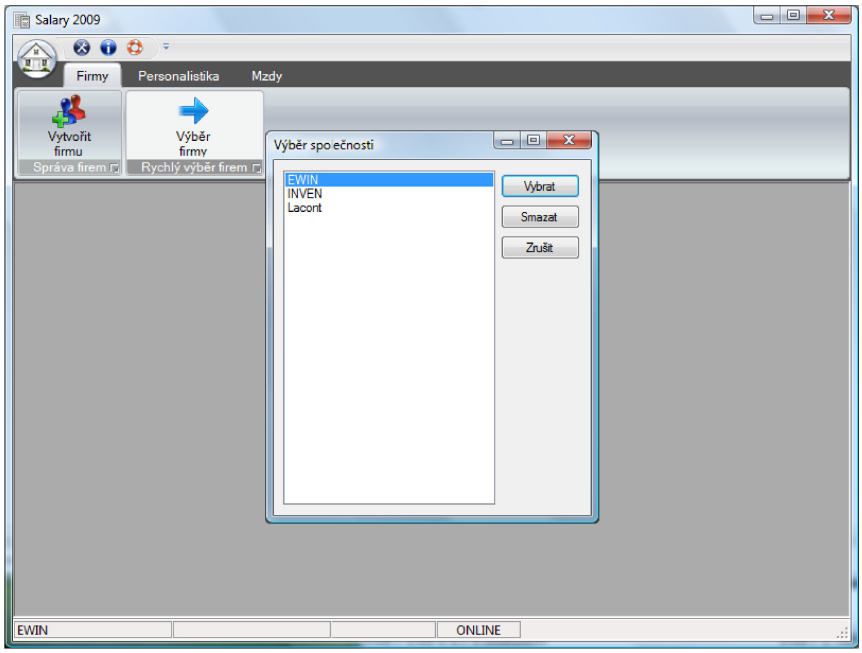
Popis diagramu:

- *SalaryContext* – spravuje objekt *SalaryStrategy*
- *SalaryStrategy* – spravuje výběr výpočtu mzdy, v diagramu naznačeno pro *CommonSalary2009* a *CommonSalary2010*
- *CommonSalary* – třída implementující rozhraní *ICommonSalary*, zabezpečující celý výpočet
- *SuperGrossSalary* – počítá superhrubou mzdu spolu s příslušným zaokrouhlením
- *GrossSalary* – počítá hrubou mzdu ze zadaných příjmů

- *HealthInsurance* – počítá zdravotní pojištění pro zaměstnance i zaměstnavatele
- *SocialInsurance* – počítá sociální pojištění pro zaměstnanci i zaměstnavatele
- *PartTax* – počítá ze zadaných vstupů dílčí základ daně z příjmů

7.4 Uživatelské rozhraní

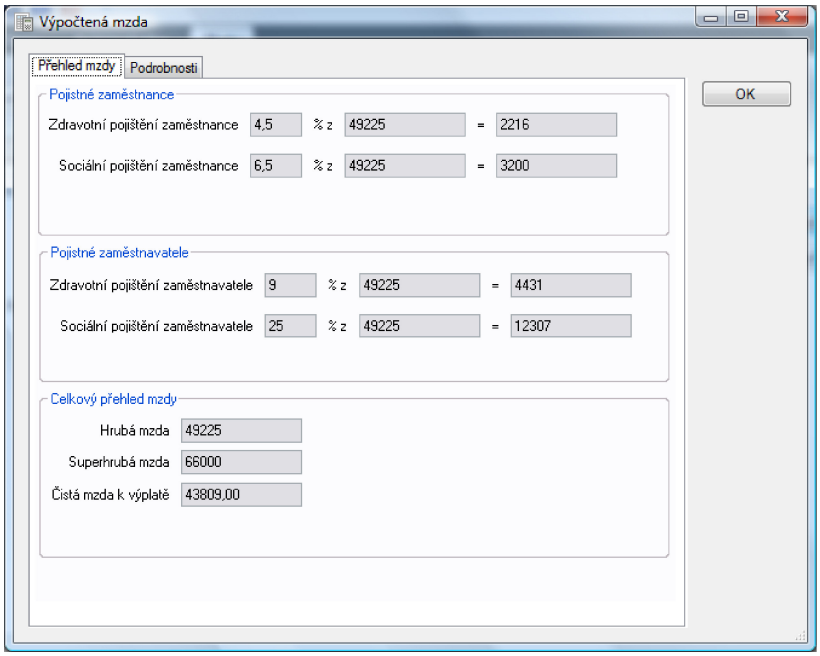
Při návrh uživatelského rozhraní *Obr. 10* bylo třeba brát v úvahu preference uživatelů. Vycházelo se z předpokladu, že většina uživatelů aktivně používá, nebo již v minulosti přišla do styku s kancelářským balíkem Microsoft Office, a tudíž by jim prostředí s ovládacím pruhem (dále ribbon [RIB09]) nemělo připadat neznámé. Ovládací prvek ribbon byl použit z volně dostupného zdroje [RIBC4].



Obr. 10: Uživatelské rozhraní

V ovládacím pruhu se nacházejí položky:

- *Firmy* – slouží pro vytvoření, výběr a správu firmy
- *Personalistika* – obdobná funkčnost správy zaměstnanců
- *Mzdy* – zadává se počet odpracovaných hodin, popřípadě vyrobených kusů



Obr. 11: Okno s vypočtenou mzdou

Po zadání všech nutných vstupních údajů se mzda vypočte a zobrazí formulář s přehledem (viz *Obr. 11*)

7.5 Testování

V průběhu vývoje byl kladen důraz na správnost kódu, nelze si totiž dovolit sebemenší chybu ve výpočtu, protože se zde manipuluje s reálnými finančními prostředky. Bylo použito jednotkových testů pomocí NUnit [NUT25] a finální kontroly mzdové účetní.

Testování bylo prováděno na různých systémech (Windows XP 32bit, Windows Vista 64bit) pro ověření funkčnosti při různých konfiguracích.

7.6 Použité programové nástroje

Při implementaci programové části jsem použil několik nástrojů třetích stran. Většina těchto nástrojů je pod open source licencí a primárně používaných pod operačním systémem Linux. Jedná se o součást repositářů majoritních distribucí, proto zde existuje menší riziko nekompatibility verzí souborů pro další pokračování ve vývoji.

7.6.1 Vývojové nástroje a prostředky

Pro návrh diagramů databáze, případu užití a různá schémata se velmi osvědčily dva programy DIA [DIA07] a StarUML [UML05]. StarUML se snaží suplovat Rational Rose, ovšem v open source podání. Nevýhoda spočívá v dostupnosti pouze pro operační systém Windows a neaktuálnosti (nejnovější verze je z roku 2005). Proto byla stěžejní část diagramů navrhnuta v programu DIA, který není tolik uživatelsky přívětivý, zato je stále aktivně vyvíjen. Program DIA není primárně určen k navrhování diagramu datových tabulek a vazeb mezi nimi, ale při použití skriptu tedia2sql [TEDIA], který z výstupního souboru vytvoří SQL skript, se dá jednoduše navrhnout databázová struktura. Největší přínos tohoto řešení bych viděl ve snadném editování diagramu, velké názornosti a přehlednosti oproti čistě textovému zápisu.

Tedia2sql dokonce podporuje různé databázové poskytovatele jako je MySQL, Sybase, Oracle, MSSQL a další. Já jsem ho použil pro generování struktury datových tabulek PostgreSQL. Pokud by někomu nestačily základní možnosti skriptu, může si ho jednoduše upravit, protože je šířen pod licencí GNU. Této možnosti jsem využil a drobnými úpravami jsem dosáhl generování výsledných SQL skriptů podle mých představ. Vygenerované SQL skripty a databázový diagram je součástí *přílohy A*.

Vývojové prostředí se mi nejvíce osvědčilo Visual Studio, přestože jsem v raných fázích uvažoval o portaci na platformu Linux a použití tamějších vývojových nástrojů jako je MonoDevelop [MDEV0]. Po rozhodnutí použít Entity Framework, který je obsažen až v nejnovější verzi .NET Frameworku, tato úvaha vzala za své. Jakmile bude implementována funkčnost .NET 3.5 SP1 do runtime prostředí MONO [MN09], tak by neměl být problém spustit modul mzdy na všech majoritních operačních systémech.

7.6.2 Databázové úložiště

Jakožto databázový zdroj jsem zvolil PostgreSQL [PGS03], které poskytuje přístupové rozhraní jak pro programovací jazyk C, C++, tak i C#, jež byl zvolen implementačním. Pro přímý přístup k databázovému úložišti je vždy nejlepší použít oficiálního poskytovatele, ať už z pohledu optimalizace přístupu, tak hlavně z možnosti použití databázově závislých vlastností, které pomocí ODBC poskytovatelů nejsou tak snadno přístupné. Dále použití oficiálního poskytovatele zaručuje přímý přístup, kdy není třeba volat abstraktní vrstvu databáze zabezpečující použití s velkou škálou datových úložišť. Mnou zvolený poskytovatel Npgsql [NPG09] ve verzi 2.0.0.4 je první verzí, která v základu poskytuje možnost snadného napojení na Entity Framework. Problém ovšem zůstává v nutnosti doinstalování poskytovatele do tzv. Global Assembly Cache, kde se sdružují programy přeložené pro .NET platformu. Dále je nutné ručně editovat soubor machine.config, který udržuje

seznam všech použitelných assembly pro přístup k databázi. V další verzi se pracuje na snadnějším způsobu instalace do systému a užším provázání se vývojářským nástrojem Visual Studio. Tuto vlastnost bych uvítal nejvíce, ať už z důvodu rychlejšího testování, tak i jednoduššího nastavení pro uživatele programu, kteří jsou nyní nuceni ručně editovat systémový soubor. Existuje zde reálné riziko neodborného zásahu do souboru a tudíž i nefunkčnosti programů, které tento soubor používají.

7.6.3 ORM nástroj

Pro objektově relační modelování jsem zvolil Entity Framework (součást .Net frameworku 3.5 SP1) a nástroje EdmGen2 [EDM02] generujícího datovou abstrakci použitou při lokální práci s databází Entity Frameworkem. Jako je například využití znalosti struktury databáze, jednotlivých datových položek, jejich typů pro testování validnosti SQL dotazů volaných nad databází.

8 Závěr

Cílem bakalářské práce bylo vhodnými nástroji modelovat a implementovat základní část mzdového modulu. Implementačním jazykem se stal C#, jako datové úložiště posloužila RDBS PotgreSQL. Vývojové nástroje byly voleny s ohledem na snadnost implementace a minimální finanční náklady, pokud možno s otevřeným zdrojovým kódem. Během vývoje jsem narazil na několik překážek v převážné většině týkajících se integrace modulů pro přístup k databázi společně s entity frameworkem, které ale nebyly fatální. Lze tedy usoudit, že spojení relativní novinky na poli objektově relačního mapování spolu s open source datovým úložištěm PostgreSQL v rámci vícevrstvé aplikace je možný a v praxi uplatnitelný. Na vhodně zvoleném vzorku dat byly vypočtené mzdy korektní, lze tedy usoudit, že průběh výpočtu je správně implementován.

Práce bude dále použita jako součást informačního systému správy mezd pro menší a středně velké firmy. Dalšími body rozšíření jsou:

- výpočty průměrného výdělku zaměstnance
- výstupy tiskových sestav (výplatní lístky, odvody na OSSZ)
- evidenční listy důchodového pojištění
- výpočet nemocenské
- automatické srážky ze mzdy
- roční vyúčtování daně
- vytváření platebních příkazů

Literatura

[ADO05] PRICE, Jason. *C# : programování databází*. 1. vyd. Praha: Grada, 2005. 642 s. ISBN 80-247-0982-1.

[CDP08] BISHOP, Judith. *C# 3.0 Design Patterns.* 1st edition. Sebastopol (USA): O'Reilly, 2008. 314 p. ISBN 0-596-52773-X.

[DAT08] AGARWAL, V. V., HUDDLESTON, J.. *Beginning C# 2008 Databases: From Novice to Professional*. 1st edition. New York: Apress, 2008. 582 p. ISBN 1-59059-900-4.

[DIA07] LARSSON, Alexander. *DIA*. [program]. Ver. 0.97-pre3. 2009. Dostupný z <http://sourceforge.net/projects/dia-installer/>. Součást projektu Gnome.

[EDM02] Microsoft. *EdmGen2*. [program]. Ver. 1.0.0. 2008. Dostupný z <http://code.msdn.microsoft.com/EdmGen2>.

[EFP08] Wikipedia. *ADO.NET Entity Framework.* [online]. <http://en.wikipedia.org/wiki/Entity_Framework>. [cit. dne 2009-5-8].

[HWL0] Wikibooks. *List of hello world programs*. [online]. <http://en.wikibooks.org/wiki/Transwiki:List_of_hello_world_programs#CIL>. [cit. 2009-5-5].

[JDBC] SUN. *Java SE Technologies - Database*. [online]. <http://java.sun.com/javase/technologies/database/>. [cit. dne 2009-5-11].

[JPA06] BISWAS Rahul, ORT Ed. *The Java Persistence API - A Simpler Programming Model for Entity Persistence*. [online]. <http://java.sun.com/developer/technicalArticles/J2EE/jpa/>. [cit. 2009-5-5].

[MDEV0] Novel. *MonoDevelop*. [program]. Ver. 2.0. 2009. Dostupný z <http://www.monodevelop.com/>. Open source IDE pro C#.

[MN09] Novel. *Mono*. [program]. Ver. 2.4. 2009. Dostupný z <http://www.mono-project.com/>. Open source runtime prostředí .NET pro Unix-like OS.

[NET0] Wikipedia. *.NET Framework*. [online]. <http://en.wikipedia.org/wiki/.Net_framework>. [cit. 2009-5-4].

[NET08] Microsoft. *Microsoft .NET Framework 3.5 Service Pack 1*. [program]. Ver. 3.5 SP1. 2008. Dostupný z <http://www.microsoft.com/downloads/details.aspx?displaylang=cs&FamilyID=ab99342f-5d1a-413d-8319-81da479ab0d7>.

[NPG09] Npgsql Development Team. *Npgsql: .Net Data Provider for Postgresql*. [program]. Ver. 2.0.0.4. 2009. Dostupný z <http://npgsql.projects.postgresql.org/>.

[NTIER0] VENKATA, Ravula. *Multi-tier Enterprise Application Architecture*. [online]. <http://www.codeproject.com/KB/architecture/Application_Architecture.aspx>. [cit. 2009-5-9].

[NUT25] NUnit.org. *NUnit*. [program]. Ver. 2.5. 2009. Dostupný z <http://www.nunit.org/index.php?p=home>. Open source nástroj pro jednotkové testy.

[PEF09] LERMAN, Julia. *Programming Entity Framework*. 1st edition. Sebastopol (USA): O'Reilly, 2009. 828 p. ISBN 0-596-52028-X.

[PGS03] MOMJIAN, Bruce. *PostgreSQL: Praktický průvodce*. 1. vyd. Brno: Computer Press, 2003. 396 s. ISBN 80-7226-954-2.

[PGS83] PostgreSQL Global Development Group. *PostgreSQL*. [program]. Ver. 8.3. 2009. Dostupný z <http://www.postgresql.org/>. Open source databázové úložiště.

[PGU09] PostgreSQL Global Development Group. *PostgreSQL Featured Users*. [online]. <http://www.postgresql.org/about/users>. [cit. 2009-5-9].

[RIB09] Wikipedia. *Ribbon (computing)*. [online]. <http://en.wikipedia.org/wiki/Ribbon_(computing)>. [cit. dne 2009-5-10].

[RIBC4] MENÉNDEZ POÓ, José. *A Professional Ribbon*. [ovládací prvek winforms]. Ver. 0.4. 2009. Dostupný z <http://www.codeproject.com/KB/toolbars/WinFormsRibbon.aspx>.

[SQL09] Microsoft. *Entity SQL Reference*. [online]. <http://msdn.microsoft.com/en-us/library/bb387118.aspx>. [cit. 2009-5-8].

[TEDIA] ELLIS, Tim. *tedia2sql*. [program]. Ver. 1.2.8. . Dostupný z <http://tedia2sql.tigris.org/>.

[UC05] COCKBURN, Alistair. *Use Cases: Jak efektivně modelovat aplikace*. 1. vyd. Brno: Computer Press, 2005. 264 s. ISBN 80-251-0721-3.

[UML05] AGORA. *StarUML*. [program]. Ver. 5.0. 2005. Dostupný z <http://staruml.sourceforge.net/en/index.php>. Open source náhrada Rational Rose.

[VIS08] Microsoft. *Visual Studio Express*. [online]. Ver. 9.0. 2008. Dostupný z <http://www.microsoft.com/Express/>.

[ZAK0] Parlament České republiky. *Zákon č. 262/2006 Sb., zákoník práce*. Ministerstvo vnitra, Praha: Tiskárna Ministerstva vnitra, 2006. 128 s. ISBN 1211-1244. Ve znění pozdějších předpisů.

Seznam příloh

Příloha A

Případy užití

Příloha B

DVD se zdrojovými kódy projektu ve formátu Visual Studio 2008, dokumentace zdrojového kódu, programová nápověda, .NET Framework 3.5 SP1, PostgreSQL 8.0.4, SQL skript pro vytvoření databáze, vzorová data pro její naplnění, text práce v elektronické podobě

Příloha C

ER diagram

Příloha D

Schéma relační databáze

Příloha A: Případy užití

Případ užití 1: Výpočet celkového úhrnu příjmů pro základ výsledné mzdy

1. **Stručný popis**
   Základ mzdy podle druhu vyplácení mzdy (fixní, hodinová, úkolová).
   1.1 **Aktéři**
   mzdová účetní
   1.2 **Spouštěče**
   Při spuštění výpočtu celkové mzdy a všech odvodů.
2. Tok událostí
   2.1 **Základní tok**
   2.1.1. Při hodinové mzdě mzdová účetní podle docházky doplní odpracované hodiny.
   2.1.1.1. Systém podle zadané hodnoty hodinové sazby vypočte hrubou mzdu.

   2.1.2. Při úkolové mzdě mzdová účetní doplní počet vyrobených kusů výrobků.
   2.1.2.1. Systém podle zadané hodnoty jednoho výrobku vypočte hrubou mzdu.

   2.1.3. Při fixní mzdě systém vypočte hrubou mzdu ze zadané fixní částky.
   2.1.4. Systém vypočte základ mzdy podle smluvního, časového nebo úkolového základu mzdy.
   2.1.5. Systém podle zadaných údajů připočte do úhrnu příjmů:
   - životní pojištění placené firmou
   - penzijní připojištění placené firmou
   - prémie
   - přesčasy
   - příplatky

   2.2 **Alternativní tok**
   2.2.1. Při zadaní hodinové mzdě a fondu pracovních hodin systém automaticky vypočítá hrubou mzdu.

3. **Speciální požadavky**

4. **Vstupní podmínky**
   Vyplněná evidence docházky u všech typů mezd.

5. **Výstupní podmínky**

6. **Body rozšíření**

Případ užití 2: Výpočet nemocenských dávek

1. **Stručný popis**
   Zaměstnanec je dočasně práce neschopný. První 3 dny mu nejsou proplaceny, další pracovní dny do 14 dne nemoci proplácí zaměstnavatel, poté sociální pojišťovna. Systém podle známých tabulek vypočítá náhradu mzdy.
   1.1 **Aktéři**
   Mzdová účetní
   1.2 **Spouštěče**
   Zaměstnanec donese hlášení o pracovní neschopnosti mzdové účetní

2. **Tok událostí**
   2.1 **Základní tok**
      2.1.1. Mzdová účetní zadá počátek nemoci do systému
      2.1.2. Mzdová účetní zadá ukončení nemoci
      2.1.3. Systém vypočte příslušnou náhradu mzdy

   2.2 **Alternativní tok**
      2.2.1. Při chybějícím dokladu o nemoci vede systém neomluvenou absenci

3. **Speciální požadavky**
   Tabulka s maximální a minimální hodnotou nemocenské.

4. **Vstupní podmínky**
   Zaměstnanec je veden v personalistice

5. **Výstupní podmínky**

6. **Body rozšíření**

Případ užití 3: Roční vyúčtování daně

1. **Stručný popis**
   Poplatník, který ve zdaňovacím období pobíral příjmy ze závislé činnosti a z funkčních požitků, může po skončení zdaňovacího období (kalendářního roku) požádat plátce daně o provedení ročního vyúčtování.
   1.1 **Aktéři**
   Mzdová účetní
   1.2 **Spouštěče**
   Zaměstnanec požádá o provedení vyúčtování
2. **Tok událostí**
   2.1 **Základní tok**
   2.1.1. Mzdová účetní vybere s systému zaměstnance.
   2.1.2. Mzdová účetní z již vypočtených mezd vybere ty, které patří do zdaňovacího období.
   2.1.3. Systém vypočte roční vyúčtování daně.

   2.2 **Alternativní tok**
   2.2.1. V systému nejsou všechny potřebně vypočtené mzdy, systém si vyžádá doplnění

3. **Speciální požadavky**
   Zadaný rozsah vyúčtování daně.

4. **Vstupní podmínky**
   Zaměstnanec je veden v personalistice

5. **Výstupní podmínky**

6. **Body rozšíření**

Případ užití 4: Vytváření platebních příkazů

1. **Stručný popis**
   Po vypočtení všech položek mzdy, je zaměstnavatel povinen tyto částky ve stanovené lhůtě uhradit. Systém vytvoří vzor platebních příkazů s doplněnými částkami a čísly účtů pro jednotlivé zaměstnance, kteří požadují úhradu na účet. Dále doplní součet částek pro OSSZ, daňový úřad a zdravotní pojišťovny.

   1.1 **Aktéři**
   Mzdová účetní.

   1.2 **Spouštěče**
   Vypočteny všechny mzdy.

2. **Tok událostí**

   2.1 **Základní tok**

   2.1.1. Mzdová účetní zadá potřebné údaje pro výpočet mezd všem zaměstnancům zadané firmy.

   2.1.2. Mzdová účetní vybere ze vzoru pro jednotlivé banky platební příkaz.

   2.1.3. Systém doplní součet příslušných vypočtených částek.

   2.1.4. Systém umožní vytištění vytvořeného příkazu.

   2.2 **Alternativní tok**

3. **Speciální požadavky**

4. **Vstupní podmínky**
   Vypočteny všechny mzdy pro zaměstnance.

5. **Výstupní podmínky**
   Platný příkaz k úhradě vytištěný dle vzoru zadané banky.

6. **Body rozšíření**